58093 String Processing Algorithms (Autumn 2015)
Exercises 3 (Tuesday, November 10)

1. Describe how to modify the LSD radix sort algorithm to handle strings of varying lengths. The time complexity should be the one given in Theorem 1.27.

2. Use the lcp comparison technique to modify the standard insertion sort algorithm so that it sorts strings in $\mathcal{O}(\Sigma LCP(\mathcal{R}) + n^2)$ time.

3. Give an example showing that the worst case time complexity of string binary search without precomputed lcp information is $\Omega(m \log n)$.

4. Let $S[0..n)$ be a string over an integer alphabet. Show how to build a data structure in $\mathcal{O}(n)$ time and space so that afterwards the Karp–Rabin hash function $H(S[i..j))$ for the factor $S[i..j)$ can be computed in constant time for any $0 \leq i \leq j \leq n$.

5. $\Omega(\Sigma LCP(\mathcal{R}))$ is a lower bound for string sorting for any algorithm if characters can be accessed only one at a time. However, for a small alphabet, it is possible to pack several characters into one machine word. Then multiple characters can be accessed simultaneously and treated as if they were a single *super-character*. For example, the string `abbaba` over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ can be thought of as the string $(\mathtt{ab}, \mathtt{ba}, \mathtt{ab})$ over the alphabet Σ^2. Algorithms taking advantage of this are called *super-alphabet* algorithms.

   Develop a super-alphabet version of MSD radix sort. What is the time complexity?